In the late 1970s, the concept of LP-optimization was introduced. Early results showed that LP optimization could be applied to many combinatorial problems. In the 1980s, a breakthrough in LP-optimization algorithms occurred, leading to efficient solutions for certain problems. In the 1990s, SDP optimization was introduced, which provided even stronger approximations for many problems. In the 2000s, new algorithms were developed, further improving the efficiency of LP and SDP methods. Today, these techniques are widely used in various fields, including computer science, operations research, and combinatorial optimization.
Def: An algo $A$ gives $\lambda$-approx for a problem $T$ if for all inputs $x$:

$\lambda T(x) \leq A(x) \leq T(x)$  where $T$ is a maximization problem

$T(x) \leq A(x) \leq \lambda^{-1} T(x)$  where $T$ is a minimization problem

Other notions:
- additive approx $\pm \beta$
- expected approx factor $E[A(x)]$

Examples

- **Partition** Given $S_1, \ldots, S_n$ partition $A \cup B = \{1, \ldots, n\}$ to min

  $\max \{ \sum_{i \in A} S_i, \sum_{j \in B} S_j \}$

  **Thm:** For any $\epsilon > 0$, there is an efficient $(1+\epsilon)$-approx algo for Partition ("polynomial time approx scheme").

  The algo picks the $\frac{1}{2} \epsilon$ largest numbers, partitions them optimally. Partitions rest greedily.

- **Clique** Given $G = (V,E)$ find the largest subset of vertices s.t. every two vertices have an edge.

  **Thm:** There is an efficient $\frac{\log n}{n}$-approx algo for clique.
- **Vertex Cover** Given $G=(V,E)$ find the smallest subset of vertices that touches all edges.

Then there is an efficient 2-approx for vertex cover.

**Pf** Pick an edge, take both its endpoints to cover, remove all covered edges.

- **Set Cover** Given $S_1, S_2, \ldots, S_m \subseteq U$, $|U|=n$, find smallest family of sets that covers $U$.

Then there is an efficient $\ln^* n$-approx for set-cover.

**Pf** Pick set that covers as many elements of $U$ as possible, remove all covered elements.

**Lem** If the opt cover has $k$ sets then in each iteration there must be a set that covers $\frac{1}{k}$ fraction of remaining elements.

Hence, if $U_i$ = uncovered elements in iteration $i$.

$$|U_i| \leq (1-\frac{1}{k})^i |U| \quad \text{when} \quad i > k \ln |U|, \quad |U_i| < 1.$$
Max 3SAT

Given clauses $C_1$--$C_m$ each of the form

$$(1) x_i \lor (2) x_j \lor (3) x_k$$

find an assignment to variables $x_1$--$x_n$ that satisfies as many clauses as possible.

Thm. There is a $3/8$-approx algo for Max 3SAT

Lem. The expected number of satisfied clauses if we pick the assignment at random is $3/8 m$.

Can derandomize using the method of conditional expectation.

Can compute efficiently the expected fraction of sat clauses; their average should be $3/8$, continue with the higher